Chapter 5
SNMP Remote Operations

An *SNMP remote oper ation* is any process on the router that can be controlled remotely using SNMP. The JUNOS software currently provides support for two SNMP remote operations: the ping MIB and traceroute MIB, defined in RFC 2925. Using these MIBs, an SNMP client in the network management system (NMS) can:

Start a series of operations on a router

Receive notification when the operations are complete

Gather the results of each operation

The JUNOS software also provides extended functionality to these MIBs in the Juniper Networks enterprise-specific extensions jnxPingMIB and jnxTraceRouteMIB. For more information about jnxPingMIB and jnxTraceRouteMIB, see “Juniper Networks Enterprise-Specific MIBs” on page 53.

This chapter covers the following topics:

SNMP Remote Operation Requirements on page 33

Use the Ping MIB on page 36

Use the Traceroute MIB on page 44

SNMP Remote Operation Requirements

To use SNMP remote operations, you should be experienced with SNMP conventions. You must also configure the JUNOS software to allow the use of the remote operation MIBs.

To configure the JUNOS software for remote operations, complete the following tasks:

Set SNMP Views on page 34

Set Trap Notification for Remote Operations on page 35

Use Variable Length String Indexes on page 35

Enable Logging on page 36

Set SNMP Views

All remote operation MIBs supported by the JUNOS software require that the SNMP clients have read-write privileges. The default SNMP configuration of the JUNOS software does not provide clients with a community string with such privileges.

To set read-write privileges for an SNMP community string, include the following statements at the [edit snmp] hierarchy level:

```
snmp {
    view view-name;
        oid object-identifier (include | exclude)
    }
    community community-name {
        authorization authorization;
        view view-name;
    }
}
```

Example: Set SNMP Views

To create a community named remote-community that grants SNMP clients read-write access to the ping MIB, jnxPing MIB, traceroute MIB, and jnxTraceRoute MIB, include the following statements at the [edit snmp] hierarchy level:

```
snmp {
    view remote-view {
        oid .1.3.6.1.2.1.80 include;            # pingMIB
        oid .1.3.6.1.4.1.2636.3.7 include;      # jnxPingMIB
        oid .1.3.6.1.2.1.81 include;            # traceRouteMIB
        oid .1.3.6.1.4.1.2636.3.8 include;      # jnxTraceRouteMIB
    }
    community remote-community {
        view remote-view;
        authorization read-write;
    }
}
```

For more information on the community statement, see “Configure the SNMP Community String” on page 21 and “community” on page 95.

For more information on the view statement, see “Configure MIB Views” on page 27 and “view” on page 107.

Set Trap Notification for Remote Operations

In addition to configuring the remote operations MIB for trap notification, you must also configure the JUNOS software. You must specify a target host for remote operations traps.

To configure trap notification for SNMP remote operations, include the categories and targets statements at the [edit snmp trap-group] hierarchy level:

```
snmp
    trap-group group-name {
        categories category;
        targets {
            address;
        }
    }
}
```

Example: Set Trap Notification for Remote Operations

Specify 172.17.12.213 as a target host for all remote operation traps:

```
snmp {
    trap-group remote-traps {
        categories remote-operations;
        targets {
            172.17.12.213;
        }
    }
}
```

For more information on trap groups, see “Configure SNMP Trap Groups” on page 25.

Use Variable Length String Indexes

All tabular objects in the remote operations MIBs supported by JUNOS are indexed by two variables of type SnmpAdminString. For more information on SnmpAdminString, see RFC 2571.

JUNOS does not handle SnmpAdminString any differently from the octet string variable type. However, the indexes are defined as variable length. When a variable length string is used as an index, the length of the string must be included as part of the OID.

Example: Set Variable Length String Indexes

To reference the pingCtlTargetAddress variable of a row in pingCtlTable where pingCtlOwnerIndex is bob and pingCtlTestName is test, use the following OID:

```
pingMIB.pingObjects.pingCtlTable.pingCtlEntry.pingCtlTargetAddress.”bob”.”test”
.1.3.6.1.2.1.80.1.2.1.4.3.98.111.98.4.116.101.115.116
```

For more information on the definition of the ping MIB, see RFC 2925.

Enable Logging

The SNMP error code returned in response to SNMP requests can only provide a generic description of the problem. The error descriptions logged by the remote operations daemon can often provide more detailed information on the problem and help you to solve the problem faster. This logging is not enabled by default. To enable logging, include the flag general statement at the [edit snmp traceoptions] hierarchy level:

```
snmp {
    traceoptions {
        flag general;
    }
}
```

For more information on traceoptions, see “Trace SNMP Activity” on page 31.

If the remote operations daemon receives an SNMP request that it cannot accommodate, the error is logged in the /var/log/rmopd file. To monitor this log file, issue the monitor start rmopd command in operational mode of the command-line interface.

Use the Ping MIB

A ping test is used to determine whether packets sent from the local host reach the designated host and are returned. If the designated host can be reached, the ping test provides the approximate round-trip time for the packets.

RFC 2925 is the authoritative description of the ping MIB in detail and provides the ASN.1 MIB definition of the ping MIB. This section provides the necessary information to:

- Start a Ping Test on page 36
- Monitor a Running Ping Test on page 37
- Gather Ping Test Results on page 41
- Stop a Ping Test on page 43
- Ping Variables on page 43

Start a Ping Test

To start a ping test, create a row in pingCtlTable and set pingCtlAdminStatus to enabled. The minimum information that must be specified before setting pingCtlAdminStatus to enabled is:

pingCtlOwnerIndex	SnmpAdminString
pingCtlTestName	SnmpAdminString
pingCtlTargetAddress	InetAddress
pingCtlRowStatus	RowStatus

For all other values, defaults are chosen unless otherwise specified. pingCtlOwnerIndex and pingCtlTestName are used as the index, so their values are specified as part of the OID. To create a row, set pingCtlRowStatus to createAndWait or createAndGo on a row that does not already exist. A value of active for pingCtlRowStatus indicates that all necessary information has been supplied and the test can begin; pingCtlAdminStatus can be set to enabled. An SNMP Set request that sets pingCtlRowStatus to active will fail if the necessary information in the row is not specified or is inconsistent.

There are two ways to start a ping test:

Use Multiple Set PDUs on page 37

Use a Single Set PDU on page 37

Use Multiple Set PDUs

You can use multiple Set request PDUs (multiple PDUs, with one or more varbind each) and set the following variables in this order to start the test:

pingCtlRowStatus to createAndWait

All appropriate test variables

pingCtlRowStatus to active

The JUNOS software now verifies that all necessary information to run a test has been specified.

pingCtlAdminStatus to enabled

Use a Single Set PDU

You can use a single Set request PDU (one PDU, with multiple varbinds) to set the following variables to start the test:

pingCtlRowStatus to createAndGo

All appropriate test variables

pingCtlAdminStatus to enabled

Monitor a Running Ping Test

When pingCtlAdminStatus is successfully set to enabled, the following is done before the acknowledgement of the SNMP Set request is sent back to the client:

pingResultsEntry is created if it does not already exist.

pingResultsOperStatus transitions to enabled.

pingResultsTable

While the test is running, pingResultsEntry keeps track of the status of the test. The value of pingResultsOperStatus is enabled while the test is running and disabled when it has stopped.

The value of pingCtlAdminStatus remains enabled until you set it to disabled. Thus, to get the status of the test, you must examine pingResultsOperStatus.

The pingCtlFrequency variable can be used to schedule many tests for one pingCtlEntry. After a test ends normally (you did not stop the test) and pingCtlFrequency number of seconds has elapsed, the test is started again just as if you had set pingCtlAdminStatus to enabled. If you intervene at any time between repeated tests (you set pingCtlAdminStatus to disabled or pingCtlRowStatus to notInService), the repeat feature is disabled until another test is started and ends normally. A value of 0 for pingCtlFrequency indicates this repeat feature is not active.

pingResultsIpTgtAddr and pingResultsIpTgtAddrType are set to the value of the resolved destination address when the value of pingCtlTargetAddressType is dns. When a test starts successfully and pingResultsOperStatus transitions to enabled:

pingResultsIpTgtAddr is set to null-string.

pingResultsIpTgtAddrType is set to unknown.

pingResultsIpTgtAddr and pingResultsIpTgtAddrType are not set until pingCtlTargetAddress can be resolved to a numeric address.To retrieve these values, poll pingResultsIpTgtAddrType for any value other than unknown after successfully setting pingCtlAdminStatus to enabled.

At the start of a test, pingResultsSentProbes is initialized to 1 and the first probe is sent. pingResultsSentProbes increases by 1 each time a probe is sent.

As the test runs, every pingCtlTimeOut seconds, the following occurs:

If a reply for the last probe sent has not been received:

pingProbeHistoryStatus for the corresponding pingProbeHistoryEntry in pingProbeHistoryTable is set to requestTimedOut.

A pingProbeFailed trap is generated, if necessary.

An attempt is made to send the next probe.

Note

No more than one outstanding probe exists for each test.

For every probe, you can receive one of the following results:

- The target host acknowledges the probe with a response.
- The probe times out; there is no response from the target host acknowledging the probe.
- The probe could not be sent.

Each probe result is recorded in pingProbeHistoryTable. For more information on pingProbeHistoryTable, see pingProbeHistoryTable on page 40.

When a response is received from the target host acknowledging the current probe:

- pingResultsProbeResponses increases by 1.
- The following variables are updated:
  - pingResultsMinRtt—Minimum round-trip time.
  - pingResultsMaxRtt—Maximum round-trip time.
  - pingResultsAverageRtt—Average round-trip time.
  - pingResultsRttSumOfSquares—Sum of squares of round-trip times.
  - pingResultsLastGoodProbe—Timestamp of the last response.

Note

Only probes that result in a response from the target host contribute to the calculation of the round-trip time (rtt) variables.

When a response to the last probe is received or the last probe has timed out, the test is complete.

pingProbeHistoryTable

An entry in pingProbeHistoryTable (pingProbeHistoryEntry) represents a probe result and is indexed by three variables:

The first two variables, pingCtlOwnerIndex and pingCtlTestName, are the same ones used for pingCtlTable, which identifies the test.

The third variable, pingProbeHistoryIndex, is a counter to uniquely identify each probe result.

The maximum number of pingProbeHistoryTable entries created for a given test is limited by pingCtlMaxRows. If pingCtlMaxRows is set to 0, no pingProbeHistoryTable entries will be created for that test.

Each time a probe result is determined, a pingProbeHistoryEntry is created and added to pingProbeHistoryTable. pingProbeHistoryIndex of the new pingProbeHistoryEntry is 1 greater than the last pingProbeHistoryEntry added to pingProbeHistoryTable for that test. pingProbeHistoryIndex is set to 1 if this is the first entry in the table. The same test can be run multiple times, so this index keeps growing.

If pingProbeHistoryIndex of the last pingProbeHistoryEntry added is 0xFFFFFFFF, the next pingProbeHistoryEntry added has pingProbeHistoryIndex set to 1.

The following is recorded for each probe result:

pingProbeHistoryResponse—Time to live (ttl)

pingProbeHistoryStatus—What happened and why

pingProbeHistoryLastRC—Return code (rc) value of ICMP packet

pingProbeHistoryTime—Timestamp when probe result was determined

When a probe cannot be sent, pingProbeHistoryResponse is set to 0. When a probe times out, pingProbeHistoryResponse is set to the difference between the time when the probe was discovered to be timed out and the time when the probe was sent.

When Traps are Generated

For any trap to be generated, the appropriate bit of pingCtlTrapGeneration must be set. A trap is generated under the following conditions:

A pingProbeFailed trap is generated every time pingCtlTrapProbeFailureFilter number of consecutive probes fail during the test.

A pingTestFailed trap is generated when the test completes and at least pingCtlTrapTestFailureFilter number of probes failed.

A pingTestCompleted trap is generated when the test completes and fewer than pingCtlTrapTestFailureFilter probes failed.

Note

A probe is considered a failure when pingProbeHistoryStatus of the probe result is anything besides responseReceived.

Gather Ping Test Results

You can either poll pingResultsOperStatus to find out when the test is complete or request to get a trap when the test is complete. For more information on pingResultsOperStatus, see pingResultsTable on page 38. For more information on ping MIB traps, see "When Traps are Generated" on page 41.

The statistics calculated and then stored in pingResultsTable include:

pingResultsMinRtt—Minimum round-trip time.

pingResultsMaxRtt—Maximum round-trip time.

pingResultsAverageRtt—Average round-trip time.

pingResultsProbeResponses—Number of responses received.

pingResultsSentProbes—Number of attempts to send probes.

pingResultsRttSumOfSquares—Sum of squares of round-trip times.

pingResultsLastGoodProbe—Timestamp of the last response.

You can also consult pingProbeHistoryTable for more detailed information on each probe. The index used for pingProbeHistoryTable starts at 1, goes to 0xFFFFFFFF, and wraps to 1 again.

For example, if pingCtlProbeCount is 15 and pingCtlMaxRows is 5, then upon completion of the first run of this test, pingProbeHistoryTable will contain probes like those in Table 3.

Table 3: Results in pingProbeHistoryTable: After the First Ping Test

pingProbeHistoryIndex	Probe Result
11	Result of 11th probe from run 1
12	Result of 12th probe from run 1
13	Result of 13th probe from run 1
14	Result of 14th probe from run 1
15	Result of 15th probe from run 1

Upon completion of the first probe of the second run of this test, pingProbeHistoryTable will contain probes like those in Table 4.

Table 4: Results in pingProbeHistoryTable: After the First Probe of Second Test

pingProbeHistoryIndex	Probe Result
12	Result of 12th probe from run 1
13	Result of 13th probe from run 1
14	Result of 14th probe from run 1
15	Result of 15th probe from run 1
16	Result of the 1st probe from run 2

Upon completion of the second run of this test, pingProbeHistoryTable will contain probes like those in Table 5.

Table 5: Results in pingProbeHistoryTable: After the Second Ping Test

pingProbeHistoryIndex	Probe Result
26	Result of 11th probe from run 2
27	Result of 12th probe from run 2
28	Result of 13th probe from run 2
29	Result of 14th probe from run 2
30	Result of 15th probe from run 2

History entries can be deleted from the MIB in two ways:

More history entries for a given test are added and the number of history entries exceeds pingCtlMaxRows. The oldest history entries are deleted to make room for the new ones.

You delete the entire test by setting pingCtlRowStatus to destroy.

Stop a Ping Test

To stop an active test, set pingCtlAdminStatus to disabled. To stop the test and remove its pingCtlEntry, pingResultsEntry, and any pingHistoryEntry objects from the MIB, set pingCtlRowStatus to destroy.

Ping Variables

This section clarifies the ranges for the following variables that are not explicitly specified in the ping MIB:

pingCtlDataSize—The value of this variable represents the total size of the payload (in bytes) of an outgoing probe packet. This payload includes the timestamp (8 bytes) that is used to time the probe. This is consistent with the definition of pingCtlDataSize (maximum value of 65507) and the standard ping application.

If the value of pingCtlDataSize is between 0 and 8 inclusive, it is ignored and the payload is 8 bytes (the timestamp). The ping MIB assumes all probes are timed, so the payload must always include the timestamp.

For example, if you wish to add an additional four bytes of payload to the packet, you must set pingCtlDataSize to 12.

pingCtlDataFill—The first 8 bytes of the data segment of the packet is for the timestamp. After that, the pingCtlDataFill pattern is used in repetition. The default pattern (when pingCtlDataFill is not specified) is (00, 01, 02, 03 ... FF, 00, 01, 02, 03 ... FF, ...). The first iteration of the default pattern starts with 08.

pingCtlMaxRows—The maximum value is 255.

pingMaxConcurrentRequests—The maximum value is 50.

pingCtlTable—The maximum number of entries allowed in this table is 100. Any attempt to create a 101st entry will result in a BAD_VALUE message for SNMPv1 and a RESOURCE_UNAVAILABLE message for SNMPv2.

pingCtlTrapProbeFailureFilter and pingCtlTrapTestFailureFilter—A value of 0 for pingCtlTrapProbeFailureFilter or pingCtlTrapTestFailureFilter is not well defined by the ping MIB. If pingCtlTrapProbeFailureFilter is 0, pingProbeFailed traps will not be generated for the test under any circumstances.If pingCtlTrapTestFailureFilter is 0, pingTestFailed traps will not be generated for the test under any circumstances.

Use the Traceroute MIB

A traceroute test approximates the path packets take from the local host to the remote host.

RFC 2925 is the authoritative description of the traceroute MIB in detail and provides the ASN.1 MIB definition of the traceroute MIB. This section provides the necessary information to:

- Start a Traceroute Test on page 44
- Monitor a Running Traceroute Test on page 45
- Monitor Traceroute Test Completion on page 49
- Gather Traceroute Test Results on page 50
- Stop a Traceroute Test on page 51
- Traceroute Variables on page 51

Start a Traceroute Test

To start a test, create a row in traceRouteCtlTable and set traceRouteCtlAdminStatus to enabled. You must specify at least the following before setting traceRouteCtlAdminStatus to enabled:

traceRouteCtlOwnerIndex	SnmpAdminString
traceRouteCtlTestName	SnmpAdminString
traceRouteCtlTargetAddress	InetAddress
traceRouteCtlRowStatus	RowStatus

For all other values, defaults are chosen unless otherwise specified. traceRouteCtlOwnerIndex and traceRouteCtlTestName are used as the index, so their values are specified as part of the OID. To create a row, set traceRouteCtlRowStatus to createAndWait or createAndGo on a row that does not already exist. A value of active for traceRouteCtlRowStatus indicates that all necessary information has been specified and the test can begin; traceRouteCtlAdminStatus can be set to enabled. An SNMP Set request that sets traceRouteCtlRowStatus to active will fail if the necessary information in the row is not specified or is inconsistent.

There are two ways to start a traceroute test:

- Use Multiple Set PDUs on page 45
- Use Single Set PDU on page 45

Use Multiple Set PDUs

You can use multiple Set request PDUs (multiple PDUs, with one or more varbind each) and set the following variables in this order to start the test:

traceRouteCtlRowStatus to createAndWait

All appropriate test variables

traceRouteCtlRowStatus to active

The JUNOS software now verifies that all necessary information to run a test has been specified.

traceRouteCtlAdminStatus to enabled

Use Single Set PDU

You can use a single Set request PDU (one PDU, with multiple varbinds) to set the following variables to start the test:

traceRouteCtlRowStatus to createAndGo

All appropriate test variables

traceRouteCtlAdminStatus to enabled

Monitor a Running Traceroute Test

When traceRouteCtlAdminStatus is successfully set to enabled, the following is done before the acknowledgement of the SNMP Set request is sent back to the client:

traceRouteResultsEntry is created if it does not already exist.

traceRouteResultsOperStatus transitions to enabled.

traceRouteResultsTable

While the test is running, this traceRouteResultsTable keeps track of the status of the test. The value of traceRouteResultsOperStatus is enabled while the test is running and disabled when it has stopped.

The value of traceRouteCtlAdminStatus remains enabled until you set it to disabled. Thus, to get the status of the test, you must examine traceRouteResultsOperStatus.

The traceRouteCtlFrequency variable can be used to schedule many tests for one traceRouteCtlEntry. After a test ends normally (you did not stop the test) and traceRouteCtlFrequency number of seconds has elapsed, the test is started again just as if you had set traceRouteCtlAdminStatus to enabled. If you intervene at any time between repeated tests (you set traceRouteCtlAdminStatus to disabled or traceRouteCtlRowStatus to notInService), the repeat feature will be disabled until another test is started and ends normally. A value of 0 for traceRouteCtlFrequency indicates this repeat feature is not active.

traceRouteResultsIpTgtAddr and traceRouteResultsIpTgtAddrType are set to the value of the resolved destination address when the value of traceRouteCtlTargetAddressType is dns. When a test starts successfully and traceRouteResultsOperStatus transitions to enabled:

traceRouteResultsIpTgtAddr is set to null-string.

traceRouteResultsIpTgtAddrType is set to unknown.

traceRouteResultsIpTgtAddr and traceRouteResultsIpTgtAddrType are not set until traceRouteCtlTargetAddress can be resolved to a numeric address. To retrieve these values, poll traceRouteResultsIpTgtAddrType for any value other other than unknown after successfully setting traceRouteCtlAdminStatus to enabled.

At the start of a test, traceRouteResultsCurHopCount is initialized to traceRouteCtlInitialTtl, and traceRouteResultsCurProbeCount is initialized to 1. Each time a probe result is determined, traceRouteResultsCurProbeCount increases by 1. While the test is running, the value of traceRouteResultsCurProbeCount reflects the current outstanding probe for which results have not yet been determined.

traceRouteCtlProbesPerHop number of probes are sent for each ttl value. When the result of the last probe for the current hop is determined, provided that the current hop is not the destination hop, traceRouteResultsCurHopCount increases by 1, and traceRouteResultsCurProbeCount resets to 1.

At the start of a test, if this is the first time this test has been run for this traceRouteCtlEntry, traceRouteResultsTestAttempts and traceRouteResultsTestSuccesses are initialized to 0.

At the end of each test execution, traceRouteResultsOperStatus transitions to disabled, and traceRouteResultsTestAttempts increases by 1. If the test was successful in determining the full path to the target, traceRouteResultsTestSuccesses increases by 1, and traceRouteResultsLastGoodPath is set to the current time.

traceRouteProbeResultsTable

Each entry in traceRouteProbeHistoryTable is indexed by five variables:

The first two variables, traceRouteCtlOwnerIndex and traceRouteCtlTestName, are the same ones used for traceRouteCtlTable and to identify the test.

The third variable, traceRouteProbeHistoryIndex, is a counter, starting from 1 and wrapping at FFFFFFFF. The maximum number of entries is limited by traceRouteCtlMaxRows.

The fourth variable, traceRouteProbeHistoryHopIndex, indicates which hop this probe is for (the actual ttl value). Thus, the first traceRouteCtlProbesPerHop number of entries created when a test starts have a value of traceRouteCtlInitialTtl for traceRouteProbeHistoryHopIndex.

The fifth variable, traceRouteProbeHistoryProbeIndex, is the probe for the current hop. It ranges from 1 to traceRouteCtlProbesPerHop.

While a test is running, as soon as a probe result is determined, the next probe is sent. A maximum of traceRouteCtlTimeOut seconds elapses before a probe is marked with status requestTimedOut and the next probe is sent. There is never more than one outstanding probe per traceroute test. Any probe result coming back after a probe times out is ignored.

Each probe can:

- Result in a response from a host acknowledging the probe
- Time out with no response from a host acknowledging the probe
- Fail to be sent

Each probe status is recorded in traceRouteProbeHistoryTable with traceRouteProbeHistoryStatus set accordingly.

Probes that result in a response from a host record the following data:

- traceRouteProbeHistoryResponse—Round-trip time (rtt)
- traceRouteProbeHistoryHAddrType—The type of HAddr (next argument)
- traceRouteProbeHistoryHAddr—The address of the hop

All probes, regardless of whether a response for the probe is received, have the following recorded:

- traceRouteProbeHistoryStatus—What happened and why
- traceRouteProbeHistoryLastRC—Return code (rc) value of the ICMP packet
- traceRouteProbeHistoryTime—Timestamp when the probe result was determined

When a probe cannot be sent, traceRouteProbeHistoryResponse is set to 0. When a probe times out, traceRouteProbeHistoryResponse is set to the difference between the time when the probe was discovered to be timed out and the time when the probe was sent.

traceRouteHopsTable

Entries in traceRouteHopsTable are indexed by three variables:

- The first two, traceRouteCtlOwnerIndex and traceRouteCtlTestName, are the same ones used for traceRouteCtlTable and identify the test.
- The third variable, traceRouteHopsHopIndex, indicates the current hop, which starts at 1 (not traceRouteCtlInitialTtl).

When a test starts, all entries in traceRouteHopsTable with the given traceRouteCtlOwnerIndex and traceRouteCtlTestName are deleted. Entries in this table are only created if traceRouteCtlCreateHopsEntries is set to true.

A new traceRouteHopsEntry is created each time the first probe result for a given ttl is determined. The new entry is created whether or not the first probe reaches a host. The value of traceRouteHopsHopIndex is increased by 1 for this new entry.

Note Any traceRouteHopsEntry can lack a value for traceRouteHopsIpTgtAddress if there are no responses to the probes with the given ttl.

Each time a probe reaches a host, the IP address of that host is available in the probe result. If the value of traceRouteHopsIpTgtAddress of the current traceRouteHopsEntry is not set, then the value of traceRouteHopsIpTgtAddress is set to this IP address. If the value of traceRouteHopsIpTgtAddress of the current traceRouteHopsEntry is the same as the IP address, then the value does not change. If the value of traceRouteHopsIpTgtAddress of the current traceRouteHopsEntry is different from this IP address, indicating a path change, a new traceRouteHopsEntry is created with:

traceRouteHopsHopIndex variable increased by 1

traceRouteHopsIpTgtAddress set to the IP address

Note A new entry for a test is added to traceRouteHopsTable each time a new ttl value is used or the path changes. Thus, the number of entries for a test may exceed the number of different ttl values used.

When a probe result is determined, the value traceRouteHopsSentProbes of the current traceRouteHopsEntry increases by 1. When a probe result is determined, and the probe reaches a host:

The value traceRouteHopsProbeResponses of the current traceRouteHopsEntry is increased by 1.

The following variables are updated:

traceRouteResultsMinRtt—Minimum round-trip time.

traceRouteResultsMaxRtt—Maximum round-trip time.

traceRouteResultsAverageRtt—Average round-trip time.

traceRouteResultsRttSumOfSquares—Sum of squares of round-trip times.

traceRouteResultsLastGoodProbe—Timestamp of the last response.

Note

Only probes that reach a host affect the round-trip time values.

When Traps are Generated

For any trap to be generated, the appropriate bit of traceRouteCtlTrapGeneration must be set.

Traps are generated under the following conditions:

traceRouteHopsIpTgtAddress of the current probe is different from the last probe with the same ttl value (traceRoutePathChange).

A path to the target could not be determined (traceRouteTestFailed).

A path to the target was determined (traceRouteTestCompleted).

Monitor Traceroute Test Completion

When a test is complete, traceRouteResultsOperStatus transitions from enabled to disabled. This transition occurs in the following situations:

The test ends successfully. A probe result indicates that the destination has been reached. In this case, the current hop is the last hop. The rest of the probes for this hop are sent. When the last probe result for the current hop is determined, the test ends.

traceRouteCtlMaxTtl threshold is exceeded. The destination is never reached. The test ends after the number of probes with ttl value equal to traceRouteCtlMaxTtl have been sent.

traceRouteCtlMaxFailures threshold is exceeded. The number of consecutive probes that end with status requestTimedOut exceeds traceRouteCtlMaxFailures.

You end the test. You set traceRouteCtlAdminStatus to disabled or delete the row by setting traceRouteCtlRowStatus to destroy.

You misconfigured the traceroute test. A value or variable you specified in traceRouteCtlTable is incorrect and will not allow a single probe to be sent. Because of the nature of the data, this error could not be determined until the test was started; that is, until after traceRouteResultsOperStatus transitioned to enabled. When this occurs, one entry is added to traceRouteProbeHistoryTable with traceRouteProbeHistoryStatus set to the appropriate error code.

If traceRouteCtlTrapGeneration is set properly, either the traceRouteTestFailed or traceRouteTestCompleted trap is generated.

Gather Traceroute Test Results

You can either poll traceRouteResultsOperStatus to find out when the test is complete or request to get a trap when the test is complete. For more information on traceResultsOperStatus, see traceRouteResultsTable on page 45. For more information on traceroute MIB traps, see "When Traps are Generated" on page 49.

Statistics are calculated on a per-hop basis and then stored in traceRouteHopsTable. They include the following for each hop:

traceRouteHopsIpTgtAddressType—Address type of host at this hop

traceRouteHopsIpTgtAddress—Address of host at this hop

traceRouteHopsMinRtt—Minimum round-trip time.

traceRouteHopsMaxRtt—Maximum round-trip time.

traceRouteHopsAverageRtt—Average round-trip time.

traceRouteHopsRttSumOfSquares—Sum of squares of round-trip times.

traceRouteHopsSentProbes—Number of attempts to send probes.

traceRouteHopsProbeResponses—Number of responses received.

traceRouteHopsLastGoodProbe—Timestamp of last response.

You can also consult traceRouteProbeHistoryTable for more detailed information on each probe. The index used for traceRouteProbeHistoryTable starts at 1, goes to 0xFFFFFFFF, and wraps to 1 again.

For example, assume the following:

traceRouteCtlMaxRows is 10.

traceRouteCtlProbesPerHop is 5.

There are 8 hops to the target (the target being number 8).

Each probe sent results in a response from a host (the number of probes sent is not limited by traceRouteCtlMaxFailures).

In this test, 40 probes are sent. At the end of the test, traceRouteProbeHistoryTable would have a history of probes like those in Table 6.

Table 6: traceRouteProbeHistoryTable

HistoryIndex	HistoryHopIndex	HistoryProbeIndex
31	7	1
32	7	2
33	7	3
34	7	4
35	7	5
36	8	1
37	8	2
38	8	3
39	8	4
40	8	5

Stop a Traceroute Test

To stop an active test, set traceRouteCtlAdminStatus to disabled. To stop a test and remove its traceRouteCtlEntry, traceRouteResultsEntry, traceRouteProbeHistoryEntry, and traceRouteProbeHistoryEntry objects from the MIB, set traceRouteCtlRowStatus to destroy.

Traceroute Variables

This section clarifies the ranges for the following variables that are not explicitly specified in the traceroute MIB:

traceRouteCtlMaxRows—The maximum value for traceRouteCtlMaxRows is 2550. This represents the maximum ttl (255) multiplied by the maximum for traceRouteCtlProbesPerHop (10). Therefore, the traceRouteProbeHistoryTable accomodates one complete test at the maximum values for one traceRouteCtlEntry. Usually, the maximum values are not used and the traceRouteProbeHistoryTable is able to accommodate the complete history for many tests for the same traceRouteCtlEntry.

traceRouteMaxConcurrentRequests—The maximum value is 50. If a test is running, it has one outstanding probe. traceRouteMaxConcurrentRequests represents the maximum number of traceroute tests that have traceRouteResultsOperStatus with a value of enabled. Any attempt to start a test with traceRouteMaxConcurrentRequests tests running will result in the creation of one probe with traceRouteProbeHistoryStatus set to maxConcurrentLimitReached and that test will end immediately.

traceRouteCtlTable—The maximum number of entries allowed in this table is 100. Any attempt to create a 101st entry will result in a BAD_VALUE message for SNMPv1 and a RESOURCE_UNAVAILABLE message for SNMPv2.